Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

優勢な配列伝達モデルは、エンコーダーとデコーダーを含む複雑なリカレントまたは畳み込みニューラルネットワークに基づいている。最も性能の良いモデルは、注意メカニズムを介してエンコーダとデコーダも接続する。我々は、再帰と畳み込みを完全に排除し、注意メカニズムのみに基づく新しいシンプルなネットワークアーキテクチャ、Transformerを提案する。2つの機械翻訳タスクの実験から、これらのモデルは並列化可能であり、学習時間が大幅に短縮される一方で、品質が優れていることが示された。我々のモデルはWMT 2014英独翻訳タスクで28.4 BLEUを達成し、アンサンブルを含む既存の最良結果を2 BLEU以上上回った。WMT 2014の英仏翻訳タスクにおいて、我々のモデルは8つのGPUで3.5日間学習した後、41.8という新しい単一モデルの最新BLEUスコアを確立した。Transformerは、大規模な学習データと限られた学習データの両方で英語の構文解析にうまく適用することで、他のタスクにうまく汎化することを示す。

* 均等な貢献。リストの順番はランダム。JakobはRNNを自己注意に置き換えることを提案し、このアイデアを評価する取り組みを開始した。AshishはIlliaと共に、最初のTransformerモデルを設計・実装し、この研究のあらゆる側面に深く関わってきた。Noamは、スケールドドットプロダクトアテンション、マルチヘッドアテンション、パラメータフリーポジション表現を提案し、ほぼすべてのディテールに関わる相手となった。Nikiは、オリジナルのcodebaseとtensor2tensorで無数のモデルバリアントを設計、実装、チューニング、評価した。Llionはまた、新しいモデルのバリエーションを実験し、最初のコードベース、効率的な推論と可視化を担当した。LukaszとAidanは、tensor2tensorの様々な部分を設計し、実装するために数え切れないほどの長日を使用し、以前のコードベースに取って代わり、結果を大幅に改善し、研究を大幅に加速した。

† Google Brain在籍時の仕事。Google Research 在籍時の仕事。

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

1 Introduction

リカレントニューラルネットワーク、特に長期短期記憶型[13]とゲーテッドリカレント[7]ニューラルネットワークは、言語モデリングや機械翻訳などのシーケンスモデリングやトランスダクション問題における最先端のアプローチとして確固たる地位を築いている[35, 2, 5]。その後、リカレント言語モデルやエンコーダ・デコーダのアーキテクチャの限界を押し広げるために、数多くの努力が続けられている[38, 24, 15]。

リカレントモデルは通常、入力シーケンスと出力シーケンスのシンボル位置に沿った計算を因数分解する。計算時間のステップに位置を合わせると、前の隠れ状態h_{t-1}と位置tの入力の関数として、隠れ状態h_tのシーケンスを生成する。このような本質的に逐次的な性質は、学習例内での並列化を妨げ、メモリ制約が例間のバッチングを制限するため、シーケンス長が長くなると重要になる。最近の研究では、因数分解のトリック[21]や条件付き計算[32]によって計算効率の大幅な改善を達成し、後者の場合のモデル性能も改善されている。しかし、逐次計算の基本的な制約は残っている。

注意メカニズムは、様々なタスクにおける説得力のあるシーケンスモデリングとトランスダクションモデルに不可欠な要素となっており、入力シーケンスや出力シーケンスにおける距離に関係なく依存関係をモデリングすることができます[2, 19]。しかし、ごく一部の例[27]を除いて、このような注意メカニズムはリカレントネットワークと組み合わせて使用される。

この研究では、再帰を避け、代わりに入力と出力の間のグローバルな依存関係を描くために、注意メカニズムに完全に依存するモデルアーキテクチャであるTransformerを提案する。Transformerは大幅に並列化を可能にし、8台のP100 GPUでわずか12時間学習した後、翻訳品質の新しい状態に到達することができる。

2 Background

逐次計算を減らすという目標は、Extended Neural GPU [16]、ByteNet [18]、ConvS2S [9]の基礎にもなっており、これらはすべて畳み込みニューラルネットワークを基本構成要素として使用し、すべての入出力位置に対して隠れ表現を並列に計算する。これらのモデルにおいて、任意の2つの入出力位置からの信号を関連付けるために必要な演算数は、位置間の距離に応じて、ConvS2Sでは線形に、ByteNetでは対数的に増加する。このため、離れた位置間の依存関係を学習することが難しくなる[12]。Transformerでは、これは一定の演算数に削減されるが、注意で重み付けされた位置を平均化することによる有効解像度の低下という代償を払うことになる。この効果は、3.2節で説明したように、マルチヘッド注意で打ち消す。

自己注意(intra-attention)とは、シーケンスの表現を計算するために、1つのシーケンスの異なる位置を関連付ける注意メカニズムである。自己注意は、読解、抽象的要約、テキスト含意、タスクに依存しない文表現の学習など、様々なタスクで成功裏に使用されている[4, 27, 28, 22]。

エンドツーエンドのメモリネットワークは、配列に沿った再帰ではなく、再帰的な注意メカニズムに基づいており、単純な言語の質問応答や言語モデリングタスクで良好な性能を発揮することが示されている[34]。

しかし、我々の知る限り、Transformerは、配列整列RNNや畳み込みを使用せずに、入力と出力の表現を計算するために完全に自己注意に依存する最初のトランスダクションモデルである。以下のセクションでは、Transformerについて説明し、自己注意を動機付け、[17, 18]や[9]のようなモデルに対する優位性について議論する。

3 Model Architecture

ほとんどの競合するニューラル配列変換モデルは、エンコーダ・デコーダ構造を持つ[5, 2, 35]。ここで、エンコーダは入力記号表現列$(x_1, \cdots, x_n)$を連続表現列$z = (z_1, \cdots, z_n)$に写像する。zが与えられると、デコーダはシンボルの出力列$(y_1, \cdots, y_m)$を1要素ずつ生成する。各ステップにおいて、モデルは自己回帰的であり[10]、以前に生成された記号を次の記号を生成する際の追加入力として消費する。

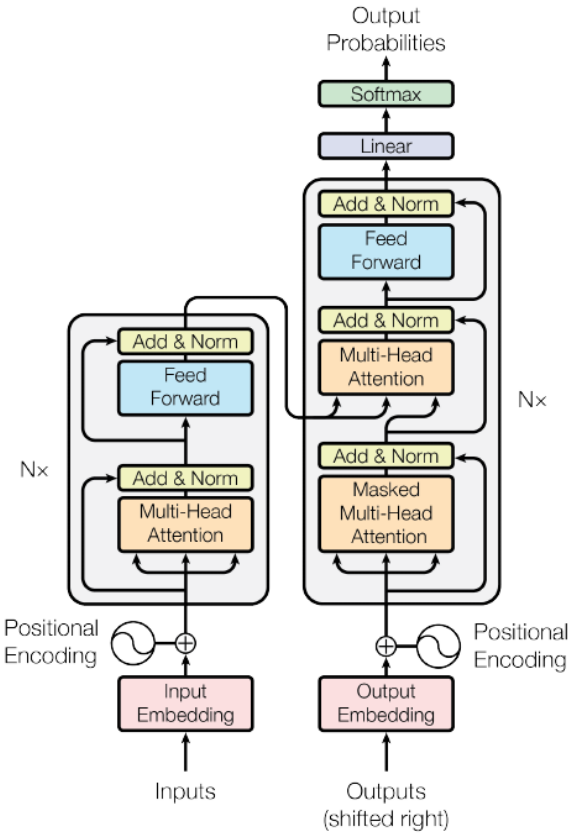

図1:Transformerモデルのアーキテクチャ。

Transformerは、図1の左半分と右半分にそれぞれ示すように、エンコーダーとデコーダーの両方に、スタックされた自己注意層とポイント単位の完全接続層を使用して、この全体的なアーキテクチャに従う。

3.1 エンコーダとデコーダのスタック

エンコーダ: エンコーダはN=6個の同一レイヤーのスタックで構成される。各層は2つのサブレイヤーを持つ。1つ目はマルチヘッド自己アテンションメカニズムで、2つ目は単純な、位置的に完全接続されたフィードフォワードネットワークである。2つのサブレイヤーのそれぞれの周りに残差接続[11]を採用し、その後レイヤーの正規化[1]を行う。すなわち、各サブレイヤーの出力はLayerNorm(x + Sublayer(x))であり、Sublayer(x)はサブレイヤー自身が実装する関数である。これらの残差接続を容易にするために、埋め込み層と同様にモデル内の全ての副層はd_{model} = 512次元の出力を生成する。

デコーダ: デコーダもN=6個の同一レイヤーのスタックで構成される。各エンコーダ層の2つのサブレイヤに加えて、デコーダは第3のサブレイヤを挿入し、エンコーダスタックの出力に対してマルチヘッドアテンションを実行する。エンコーダと同様に、各サブレイヤーの周囲に残差接続を採用し、レイヤーの正規化を行う。また、デコーダスタックの自己アテンションサブレイヤーを変更し、ポジションが後続のポジションにアテンションしないようにする。このマスキングは、出力埋め込みが1位置オフセットされることと組み合わされ、位置iの予測がiより小さい位置の既知の出力にのみ依存することを保証する。

3.2 Attention

アテンション関数は、クエリとキーと値のペアのセットを出力にマッピングするものとして記述することができ、クエリ、キー、値、および出力はすべてベクトルである。

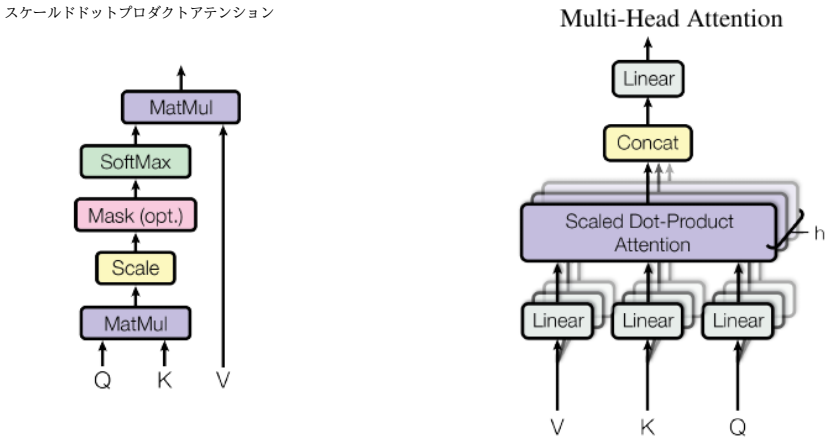

図2:(左)スケールドドットプロダクトアテンション。(右) マルチヘッドアテンションは、並列に実行される複数のアテンションレイヤーから構成される。

出力は値の加重和として計算され、各値に割り当てられた重みは、対応するキーとクエリの互換性関数によって計算される。

3.2.1 スケールドドットプロダクトアテンション

我々はこの特殊な注意を「スケールドドットプロダクト注意」と呼ぶ(図2)。入力はd_k次元のクエリとキー、およびd_v次元の値からなる。全てのキーとクエリのドット積を計算し、それぞれを$\sqrt{d_k}$で割り、ソフトマックス関数を適用して値の重みを求める。

実際には、クエリの集合を同時に行列Qに詰め込んで注目関数を計算する。また、キーと値は行列KとVにまとめて詰め込まれる。出力の行列を次のように計算する:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

最もよく使われる注意関数は、加法的注意[2]とドット積(乗法)注意の2つである。ドット積注目度は、$\sqrt{1}$のスケーリングファクターを除いて、我々のアルゴリズムと同じである。加法的注意は、d k を単一の隠れ層とするフィードフォワードネットワークを用いて互換性関数を計算する。この2つは理論的な複雑さでは似ているが、ドット積アテンションは高度に最適化された行列乗算コードを用いて実装できるため、実際にはより高速でスペース効率も高い。

d_kの値が小さい場合には、2つのメカニズムは同様の性能を示すが、d_kの値が大きい場合には、加法的注意はスケーリングせずにドット積注意を上回る[3]。d_kの値が大きいと、ドット積の大きさが大きくなり、ソフトマックス関数が極端に小さい勾配[4]を持つ領域に押し込まれるのではないかと推測される。この効果を打ち消すために、ドット積を$\sqrt{1}$でスケーリングする。d

3.2.2 マルチヘッド注意

d_{model} -次元のキー、値、クエリで単一の注意関数を実行する代わりに、クエリ、キー、値をそれぞれd_k、d_k、d_v次元に異なる学習済み線形投影でh回線形投影することが有益であることがわかった。

ドット積が大きくなる理由を説明するために、qとkの成分が平均0、分散1の独立な確率変数であると仮定する。すると、そのドット積である$q \cdot k = \sum_{i=1}^{d_k} q_i k_i$ は平均0、分散d_k となる。

これらの投影されたクエリ、キー、値のそれぞれに対して、注意関数を並列に実行し、d_v -次元の出力値を得る。これらを連結し、もう一度投影すると、図2に示すような最終的な値が得られる。

マルチヘッドアテンションにより、モデルは異なる位置の異なる表現部分空間からの情報に共同してアテンションすることができる。単一の注意ヘッドでは、平均化によってこれが阻害される。

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_\text{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_\text{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_\text{model}}$.

この研究では、h = 8個の並列注意層(ヘッド)を採用する。これらそれぞれについて、$d_k = d_v = d_{model} / h = 64$とする。各ヘッドの次元が小さくなるため、総計算コストは全次元の単一ヘッド注意と同様である。

3.2.3 我々のモデルにおける注意の応用

Transformerは3つの異なる方法でマルチヘッドアテンションを使用します:

- エンコーダ・デコーダ注目層では、クエリは前のデコーダ層から、メモリキーと値はエンコーダの出力から来る。これにより、デコーダ内のすべての位置が、入力シーケンス内のすべての位置にわたってアテンションすることができる。これは、[38, 2, 9]のようなシーケンス間モデルにおける典型的なエンコーダ・デコーダの注意メカニズムを模倣したものである。
- エンコーダは自己注意層を含む。自己アテンション層では、キー、値、クエリはすべて同じ場所、この場合はエンコーダの前の層の出力から来る。エンコーダの各位置は、エンコーダの前層のすべての位置にアテンションできる。
- 同様に、デコーダの自己アテンション層は、デコーダの各位置が、その位置までのデコーダのすべての位置にアテンションすることを可能にする。自己回帰特性を維持するために、デコーダの左向きの情報フローを防ぐ必要がある。我々は、不正な接続に対応するソフトマックスの入力のすべての値をマスキング(-∞に設定)することで、スケーリングされたドット積注意の内部でこれを実装する。図2を参照。

3.3 位置ごとのフィードフォワードネットワーク

注意のサブレイヤーに加えて、我々のエンコーダとデコーダの各レイヤーは、各位置に別々に同一に適用される完全接続のフィードフォワードネットワークを含んでいる。これは2つの線形変換とその間のReLU活性化からなる。

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2)$$

線形変換は異なる位置で同じであるが、層ごとに異なるパラメータを使用する。これを説明するもう一つの方法は、カーネルサイズ1の2つの畳み込みである。入力と出力の次元は$d_{model} = 512$であり、内層の次元は$d_{ff} = 2048$である。

3.4 埋め込みとソフトマックス

他の配列変換モデルと同様に、入力トークンと出力トークンを次元d_{model}のベクトルに変換するために学習済み埋め込みを使用する。また、通常の学習済み線形変換とソフトマックス関数を用いて、デコーダ出力を予測される次のトークンの確率に変換する。我々のモデルでは、[30]と同様に、2つの埋め込み層とプリソフトマックス√線形変換の間で同じ重み行列を共有する。埋め込み層では、これらの重みにd_{model}を乗じる.

表1:異なるレイヤータイプにおける最大パス長、レイヤーごとの複雑さ、および最小逐次演算数。nはシーケンス長、dは表現次元、kは畳み込みのカーネルサイズ、rは制限された自己注意における近傍のサイズである。

Layer Type	Complexity per Layer	Sequential Operations	最大経路長
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
自己アテンション(制限付き)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

3.5 位置エンコーディング

我々のモデルには再帰性も畳み込みもないので、モデルがシーケンスの順序を利用するためには、シーケンス内のトークンの相対位置または絶対位置に関する情報を注入しなければならない。この目的のために、エンコーダとデコーダのスタックの底にある入力埋め込みに「位置エンコーディング」を追加する。位置エンコーディングは埋め込みと同じ次元d_{model}を持つので、両者を合計することができる。位置エンコーディングには、学習型と固定型の多くの選択肢がある[9]。

本研究では、異なる周波数のサイン関数とコサイン関数を使用する:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

ここで、posは位置、iは次元である。すなわち、位置エンコーディングの各次元は正弦波に対応する。波長は2πから$10000 \cdot 2\pi$までの幾何学的な進行を形成する。任意の固定オフセットkに対して、PE_{pos+k}はPE_{pos}の一次関数として表現できるため、モデルが相対位置による出席を容易に学習できると仮定して、この関数を選択した。

また、代わりに学習済みの位置埋め込み[9]を使用する実験も行い、2つのバージョンでほぼ同じ結果が得られることがわかった(表3の行(E)参照)。正弦波バージョンを選んだのは、トレーニング中に遭遇したシーケンスよりも長いシーケンス長にモデルを外挿できる可能性があるからである。

4 Why Self-Attention

このセクションでは、自己注意層の様々な側面を、ある可変長の記号表現列($x_1, \cdots, x_n$)を、$x_i, z_i \in \mathbb{R}^d$と等長の別のシーケンス($z_1, \ldots, z_n$)にマッピングするためによく使われるリカレント層と畳み込み層、例えば典型的なシーケンス変換エンコーダやデコーダの隠れ層などと比較する。自己注意を使う動機付けとして、3つの望ましい条件を考える。

1つはレイヤーごとの総計算量である。もう一つは、並列化できる計算量であり、必要とされる逐次演算の最小数で測定される。

3つ目は、ネットワーク内の長距離依存関係間のパスの長さである。長距離依存性の学習は、多くの配列変換タスクにおける重要な課題である。このような依存関係を学習する能力に影響を与える重要な要因の1つは、前方信号と後方信号がネットワーク内で通過しなければならない経路の長さである。入力配列と出力配列の任意の位置の組み合わせの間のこれらのパスが短ければ短いほど、長距離依存性の学習が容易になります[12]。したがって、異なる層のタイプで構成されるネットワークにおいて、任意の2つの入出力位置間の最大経路長も比較する。

表1にあるように、自己アテンション層は一定数の逐次実行操作で全ての位置を接続するのに対し、リカレント層はO(n)の逐次実行操作を必要とする。計算量の点では、配列長nが表現次元dより小さい場合、

自己注意層はリカレント層より高速であり、これは機械翻訳の最先端モデルで用いられる文表現、例えばワードピース[38]やバイトペア[31]表現で最もよく見られるケースである。非常に長いシーケンスを含むタスクの計算性能を向上させるために、自己注意を、それぞれの出力位置を中心とした入力シーケンスのサイズrの近傍のみを考慮するように制限することができる。これにより、最大経路長はO(n/r)に増加する。今後の研究で、このアプローチをさらに調査する予定である。

カーネル幅 k < n の単一の畳み込み層は、入力と出力の位置のすべてのペアを接続しない。そのためには、連続カーネルの場合はO(n/k)、拡張畳み込みの場合はO($\log_k$ (n))の畳み込み層を積み重ねる必要があり[18]、ネットワーク内の任意の2つの位置間の最長パスの長さが長くなる。畳み込み層は一般にリカレント層よりk倍も高価である。しかし、分離可能な畳み込み[6]は、複雑さを大幅に減少させ、O(k - n - d + n - d^2)となる。しかし、k = nであっても、分離可能な畳み込みの複雑さは、我々のモデルで採用している自己注意層とポイントワイズフィードフォワード層の組み合わせに等しい。

副次的な利点として、自己アテンションはより解釈しやすいモデルをもたらす可能性がある。我々のモデルから注意の分布を検査し、付録で例を提示し、議論する。個々の注意の頭は明らかに異なるタスクを実行するように学習するだけでなく、多くは文の構文構造や意味構造に関連した振る舞いを示すようである。

5 Training

このセクションでは、我々のモデルの学習体制について説明する。

5.1 学習データとバッチ処理

約450万文対からなる標準的なWMT 2014英独データセットで学習を行った。文はバイトペアエンコーディング[3]を用いてエンコードされ、約37000トークンのソースターゲット語彙が共有されている。英語-フランス語については、3600万文からなる非常に大規模なWMT 2014英語-フランス語データセットを使用し、トークンを32000ワードピースの語彙に分割した[38]。文のペアは、おおよその配列の長さによってバッチ化された。各トレーニングバッチには、約25000のソーストークンと25000のターゲットトークンを含む文ペアのセットが含まれる。

5.2 ハードウェアとスケジュール

NVIDIA P100 GPUを8台搭載した1台のマシンでモデルを学習させた。本論文で説明したハイパーパラメータを用いたベースモデルでは、各トレーニングステップに約0.4秒を要した。ベースモデルの学習は、合計10万ステップ、12時間行った。表3の最下行にあるビッグモデルでは、ステップ時間は1.0秒であった。ビッグモデルは30万ステップ(3.5日)学習された。

5.3 Optimizer

Adam optimizer [20] を用い、$\beta_1 = 0.9$, $\beta_2 = 0.98$, $\varepsilon = 10^{-9}$ とした。学習の過程で学習率を変化させ、その式に従って学習を行った。

$$lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \tag{3}$$

これは、最初のwarmup_stepsの学習ステップでは学習率を直線的に増加させ、それ以降はステップ数の逆平方根に比例して減少させることに相当する。warmup_steps = 4000を使用。

5.4 Regularization

学習時に3種類の正則化を採用しています:

表2:Transformerは、英語からドイツ語、英語からフランス語のnewstest2014テストにおいて、学習コストのほんの一部で、以前の最先端モデルよりも優れたBLEUスコアを達成した。

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
変換器(基本モデル) 変換	27.3	38.1	**$3.3 \cdot 10^{18}$**	
器(大)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

残差ドロップアウト(Residual Dropout) 各サブレイヤーの出力にドロップアウト[33]を適用し、サブレイヤーの入力に追加して正規化する前に、ドロップアウトを適用する。さらに、エンコーダスタックとデコーダスタックの両方において、埋め込みと位置エンコーディングの和にドロップアウトを適用する。ベースモデルでは、P_{drop} = 0.1の割合を用いる。

ラベルスムージング ε_{ls} = 0.1 [36]のラベルスムージングを採用した。これは、モデルがより確信が持てないことを学習するため、当惑を悪化させるが、精度とBLEUスコアが向上する。

6 Results

6.1 機械翻訳

WMT 2014英独翻訳タスクにおいて、ビッグトランスフォーマーモデル(表2のTransformer(big))は、これまでに報告された最良のモデル(アンサンブルを含む)を2.0 BLEU以上上回り、28.4という新たな最先端BLEUスコアを確立した。このモデルの構成を表3の最下行に示す。8台のP100 GPUで3.5日間のトレーニングを要した。我々の基本モデルでさえ、競合モデルの学習コストのほんの一部で、以前に発表された全てのモデルとアンサンブルを凌駕している。

WMT 2014の英仏翻訳タスクにおいて、我々のビッグモデルはBLEUスコア41.0を達成し、過去に発表された全ての単一モデルを上回った。英語からフランス語への学習でTransformer(big)モデルは、ドロップアウト率P_{drop} = 0.3ではなく、0.1を使用した。

ベースモデルには、10分間隔で書かれた直近の5つのチェックポイントを平均して得られた単一のモデルを使用した。ビッグモデルについては、直近の20個のチェックポイントを平均した。ビームサイズ4、長さペナルティα=0.6[38]のビームサーチを用いた。これらのハイパーパラメータは、開発セットで実験した後に選択された。推論中の最大出力長を入力長+50に設定したが、可能な限り早期に終了させる[38]。

表2は、我々の結果を要約し、我々の翻訳品質と学習コストを、文献にある他のモデルアーキテクチャと比較したものである。学習時間、使用したGPUの数、各GPUの持続的な単精度浮動小数点演算能力の推定値[5]を乗じることで、モデルの学習に使用した浮動小数点演算の数を推定する。

6.2 モデルのバリエーション

Transformerの様々なコンポーネントの重要性を評価するために、開発セットであるnewstest2013における英独翻訳の性能の変化を測定し、

[5]We used values of 2.8, 3.7, 6.0 and 9.5 TFLOPS for K80, K40, M40 and P100, respectively.

表3:Transformerアーキテクチャのバリエーション。未記載の値はベースモデルと同じである。すべてのメトリクスは英独翻訳開発セットnewstest2013のものである。表中の当惑度はバイトペアエンコーディングによる単語単位であり、単語単位の当惑度と比較すべきではない。

	N	d_{model}	d_{ff}	h	d_k	d_v	P_{drop}	ϵ_{ls}	train steps	PPL (dev)	BLEU (dev)	params $\times 10^6$
base	6	512	2048	8	64	64	0.1	0.1	100K	4.92	25.8	65
(A)				1	512	512				5.29	24.9	
				4	128	128				5.00	25.5	
				16	32	32				4.91	25.8	
				32	16	16				5.01	25.4	
(B)					16					5.16	25.1	58
					32					5.01	25.4	60
(C)	2									6.11	23.7	36
	4									5.19	25.3	50
	8									4.88	25.5	80
		256			32	32				5.75	24.5	28
		1024			128	128				4.66	26.0	168
			1024							5.12	25.4	53
			4096							4.75	26.2	90
(D)							0.0			5.77	24.6	
							0.2			4.95	25.5	
								0.0		4.67	25.3	
								0.2		5.47	25.7	
(E)	正弦波の代わりに位置埋め込みを行う									4.92	25.7	
big	6	1024	4096	16			0.3		300K	**4.33**	**26.4**	213

ベースモデルを様々な方法で変化させた。前節で説明したようにビームサーチを使用したが、チェックポイントの平均化は行わなかった。これらの結果を表3に示す。

表3の行(A)では、セクション3.2.2で説明したように、計算量を一定に保ちながら、注意ヘッドの数、注意のキーと値の次元を変化させている。シングルヘッドの注意は最良の設定より0.9BLEU悪いが、ヘッド数が多すぎると品質も落ちる。

表3の行(B)では、注目鍵のサイズd_kを小さくすると、モデルの品質が低下することがわかる。このことは、互換性の決定が容易ではなく、ドット積よりも洗練された互換性関数が有益であることを示唆している。さらに(C)と(D)の行で、予想通り、より大きなモデルの方が優れており、ドロップアウトはオーバーフィッティングを避けるのに非常に有効であることがわかる。(E)の行では、正弦波位置エンコーディングを学習された位置埋め込み[9]に置き換えており、ベースモデルとほぼ同じ結果を観察している。

6.3 英語の構成語解析

Transformerが他のタスクに一般化できるかどうかを評価するために、英語の構文解析の実験を行った。このタスクは、出力が強い構造的制約を受け、入力よりもかなり長いという特殊な課題を提示する。さらに、RNNのsequence-to-sequenceモデルは、小さなデータ領域で最先端の結果を達成することができなかった[37]。

Penn Treebank [25]のWall Street Journal (WSJ) 部分、約40Kの学習文に対して、d_{model} = 1024の4層変換器を学習させた。また、約17Mの文からなる大規模な高信頼度コーパスとBerkleyParserコーパス[37]を用いて、半教師付き設定での学習も行った。WSJのみの設定では16Kトークンの語彙を使用し、半教師付きの設定では32Kトークンの語彙を使用した。

セクション22の開発セットで、ドロップアウト、注意と残差(セクション5.4)の両方、学習率、ビームサイズを選択するために少数の実験を行っただけで、他のすべてのパラメータは英独ベース翻訳モデルと変更しなかった。

表4:Transformerは英語の構文解析によく汎化する(結果はWSJのセクション23にある)

Parser	Training	WSJ 23 F1
Vinyals & Kaiser el al. (2014) [37]	WSJのみ、識別WSJのみ、識	88.3
Petrov et al. (2006) [29]	別WSJのみ、識別WSJのみ、	90.4
Zhu et al. (2013) [40]	識別WSJのみ、識別WSJのみ	90.4
Dyer et al. (2016) [8]		91.7
Transformer (4 layers)	WSJ only, discriminative	91.3
Zhu et al. (2013) [40]	semi-supervised	91.3
Huang & Harper (2009) [14]	semi-supervised	91.3
McClosky et al. (2006) [26]	semi-supervised	92.1
Vinyals & Kaiser el al. (2014) [37]	semi-supervised	92.1
Transformer (4 layers)	semi-supervised	92.7
Luong et al. (2015) [23]	multi-task	93.0
Dyer et al. (2016) [8]	generative	93.3

推論中、最大出力長を入力長+300に増やした。WSJのみと半教師付き設定の両方で、ビームサイズ21、α=0.3を使用した。

表4の結果から、タスクに特化したチューニングがないにもかかわらず、我々のモデルは驚くほど良い性能を示し、リカレントニューラルネットワーク文法[8]を除く全ての既報モデルよりも良い結果を得たことがわかる。

RNNのsequence-to-sequenceモデル[37]とは対照的に、Transformerは40K文のWSJ訓練セットのみで訓練した場合でも、BerkeleyParser[29]を上回る。

7 Conclusion

本研究では、エンコーダ・デコーダのアーキテクチャで最も一般的に使用されるリカレント層を、多頭の自己注意に置き換えた、完全に注意に基づく最初のシーケンス変換モデルであるTransformerを発表した。

翻訳タスクにおいて、Transformerはリカレント層や畳み込み層に基づくアーキテクチャよりも大幅に高速に学習することができる。WMT 2014英独翻訳タスクとWMT 2014英仏翻訳タスクの両方において、我々は新たな技術水準を達成した。前者のタスクでは、我々の最良のモデルは、以前に報告されたすべてのアンサンブルを凌駕した。

我々は注意に基づくモデルの将来性に期待しており、他のタスクにも適用する予定である。我々は、Transformerをテキスト以外の入出力モダリティを含む問題に拡張し、画像、音声、ビデオなどの大きな入出力を効率的に扱うための局所的で制限された注意メカニズムを研究する予定である。世代をより逐次的でなくすることも、我々の研究目標である。

モデルの学習と評価に使用したコードは、https://github.com/tensorflow/tensor2tensorで公開されている。

謝辞 Nal KalchbrennerとStephan Gouwsの有益なコメント、修正、インスピレーションに感謝する。

References

[1] Jimmy Lei Ba、Jamie Ryan Kiros、Geoffrey E Hinton。レイヤーの正規化. arXiv preprint arXiv:1607.06450, 2016.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio.整列と翻訳を共同で学習することで、ニューラル機械翻訳を実現。CoRR、abs/1409.0473、2014。

[3] デニー・ブリッツ、アンナ・ゴニー、ミン・タン・ルオン、クオック・V・レ。ニューラル機械翻訳アーキテクチャの大規模な探索。CoRR、abs/1703.03906、2017年。

[4] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[5] チョ・キョンヒョン、バート・ヴァン・メリエンボア、カグラル・グルセール、フェティ・ブーガレス、ホルガー・シュヴェンク、ヨシュア・ベンジオ。統計的機械翻訳のためのrnnエンコーダ・デコーダを用いたフレーズ表現の学習。CoRR、abs/1406.1078、2014。

[6] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.

[7] Junyoung Chung, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[8] Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. Recurrent neural network grammars. In *Proc. of NAACL*, 2016.

[9] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122v2*, 2017.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[12] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[13] Sepp HochreiterとJürgen Schmidhuber。長期短期記憶。神経計算，9(8):1735-1780，1997.

[14] 黄中強、メアリー・ハーパー。言語間の潜在的な注釈を持つPCFG文法を自己学習させる。自然言語処理における経験的方法に関する2009年会議録、ページ832-841。ACL，August 2009.

[15] ラファル・ヨゼフォヴィッチ、オリオル・ヴィニャルズ、マイク・シュスター、ノーム・シャゼール、ウー・ヨンホイ。言語モデリングの限界を探る。arXivプレプリントarXiv:1602.02410，2016.

[16] Łukasz KaiserとSamy Bengio。アクティブメモリは注意に取って代わることができるか?In Advances in Neural Information Processing Systems，(NIPS)，2016.

[17] Łukasz KaiserとIlya Sutskever。ニューラルGPUはアルゴリズムを学習する。国際学習表現会議(ICLR)，2016 にて。

[18] ナール・カルチブレナー、ラッセ・エスペホルト、カレン・シモニャン、アーロン・ヴァン・デン・オード、アレックス・グレイブス、コレイ・カヴクオグル。線形時間でのニューラル機械翻訳。arXivプレプリントarXiv:1610.10099v2，2017.

[19] ユン・キム、カール・デントン、ルオン・ホアン、アレクサンダー・M・ラッシュ。構造化された注意ネットワーク。学習表現に関する国際会議，2017 にて。

[20] ディエデリク・キングマ、ジミー・バAdam: 確率的最適化のための手法。ICLR，2015.

[21] オレクシイ・クシェフ、ボリス・ギンズバーグLSTMネットワークの因数分解トリック。arXivプレプリントarXiv:1703.10722，2017.

[22] 林周漢、峰敏偉、キケロ・ノゲイラ・ドス・サントス、モ・ユー、ビン・シャン、ボーエン・ズー、ヨシュア・ベンジオ。構造化された自己注視型文埋め込み。arXivプレプリントarXiv:1703.03130，2017.

[23] Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*, 2015.

[24]ミン・タン・ルオン、ヒョウ・ファム、クリストファー・D・マニング。注意に基づくニューラル機械翻訳への効果的なアプローチarXivプレプリントarXiv:1508.04025，2015.

[25] ミッチェル・P・マーカス、メアリー・アン・マルシンキエヴィッチ、ベアトリス・サントリーニ。英語の大規模な注釈付きコーパスの構築: pennツリーバンク。計算言語学，19(2):313-330，1993.

[26] デビッド・マクロスキー、ユージン・シャルニアック、マーク・ジョンソン。構文解析のための効果的な自己学習。NAACL人間言語技術会議本大会講演論文集，ページ152-159. ACL，June 2006.

[27] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model. In *Empirical Methods in Natural Language Processing*, 2016.

[28] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[29] スラブ・ペトロフ、レオン・バレット、ロマン・ティボー、ダン・クライン。正確でコンパクト、かつ解釈可能な木の注釈を学習する。第21回計算言語学国際会議および第44回ACL年次総会予稿集，ページ 433-440. ACL，July 2006.

[30] Ofir Press and Lior Wolf. 言語モデルを改善するための出力埋め込みの使用。arXivプレプリントarXiv:1608.05859，2016.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] ノーム・シャゼール、アザリア・ミルホセイニ、クジシュトフ・マジアルツ、アンディ・デイビス、クオック・レ、ジェフリー・ヒントン、ジェフ・ディーン。異常に大きなニューラルネットワーク: スパースゲート型専門家混合層。arXivプレプリントarXiv:1701.06538，2017.

[33] ニティシュ・スリヴァスタヴァ、ジェフリー・E・ヒントン、アレックス・クリシェフスキー、イリヤ・スーツケバー、ルスラン・サラフチノフ。ドロップアウト:ニューラルネットワークのオーバーフィッティングを防ぐ簡単な方法。機械学習研究，15(1):1929-1958，2014.

[34] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[35] イリヤ・スーツケバー、オリオル・ビニャルズ、クオックVV・レ。ニューラルネットワークによる配列間学習In Advances in Neural Information Processing Systems，pages 3104-3112，2014.

[36] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[37] Vinyals & Kaiser, Koo, Petrov, Sutskever, and Hinton. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, 2015.

[38] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[39] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, and Wei Xu. Deep recurrent models with fast-forward connections for neural machine translation. *CoRR*, abs/1606.04199, 2016.

[40] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 1: Long Papers)*, pages 434–443. ACL, August 2013.

Attention Visualizations

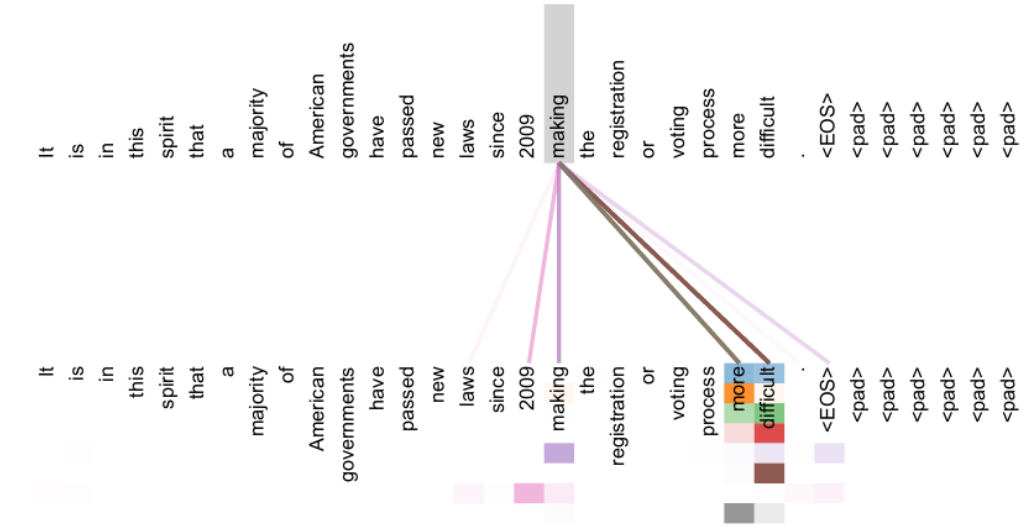

図3:第5層のエンコーダの自己注意における長距離依存性に従う注意メカニズムの例 6の第5層のエンコーダの自己注意における長距離依存性に従う注意メカニズムの例。注意ヘッドの多くは動詞「作る」の遠い依存性に注意し、「もっと作る」というフレーズを完成させる。ここでの注意は「作る」という単語に対してのみ示されている。色の違いは、異なる頭部を表している。カラーで見るのがベスト

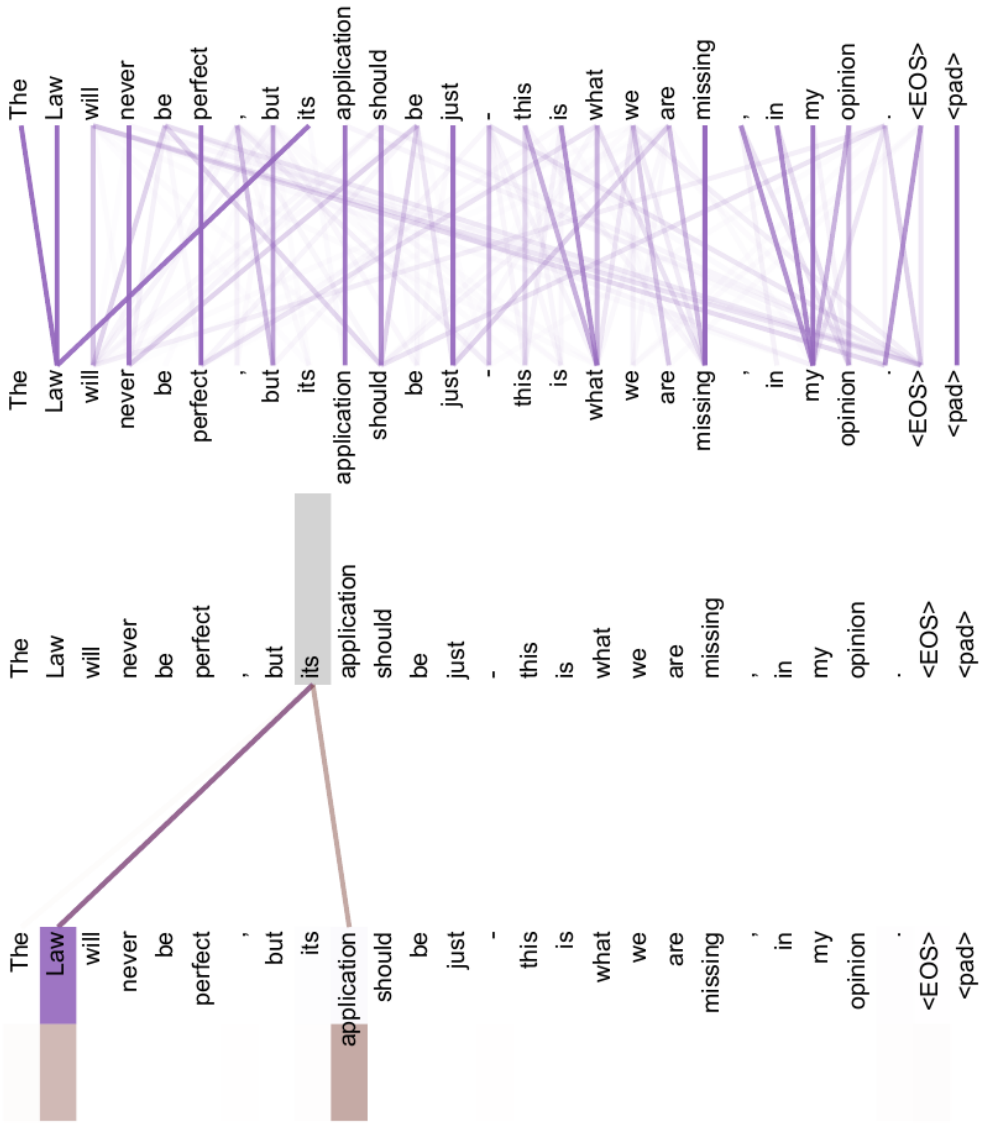

図4:同じく第6層の第5層にある、アナフォラ解決に関与していると思われる2つの注意の頭。上:頭部5に対する完全な注意。下: 注意ヘッド5と6の'its'という単語だけから分離された注意。この単語の注意は非常にシャープであることに注意。

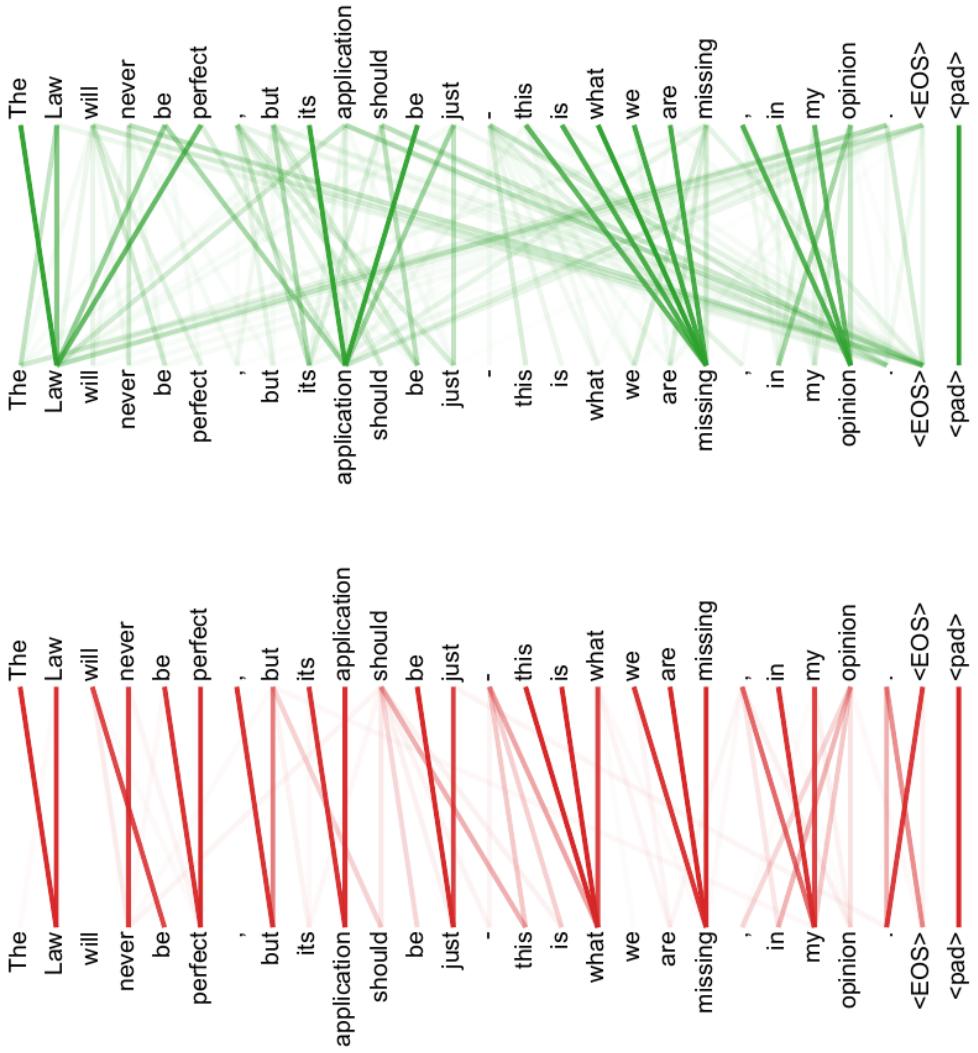

図5:注意の頭の多くは、文の構造に関連していると思われる振る舞いを示す。上記の2つの例を挙げると、6層目のエンコーダの自己アテンションから、2つの異なるヘッドからである。ヘッドは明らかに異なるタスクを実行するように学習した。